

Linear complexity SimRank computation based on the iterative diagonal estimation

I.V. Oseledets
Skolkovo Institute of Science
and Technology, Novaya St.,
100, Skolkovo, 143025,
Russia
Institute of Numerical
Mathematics, Russian
Academy of Sciences,
Gubkina St., 8, Moscow,
119333
i.oseledets@skoltech.ru

G.V. Ovchinnikov
Skolkovo Institute of Science
and Technology, Novaya St.,
100, Skolkovo, 143025,
Russia
Institute for Design Problems
in Microelectronics, Russian
Academy of Sciences, prosp.
60-letiya Oktyabrya, 9,
Moscow, Russia
ovgeorge@yandex.ru

A. M. Katrutsa
Skolkovo Institute of Science
and Technology, Novaya St.,
100, Skolkovo, 143025,
Russia
Moscow Institute of Physics
and Technology, Institutskiy
Lane 9, Dolgoprudny, 141700,
Russia
aleksandr.katrutsa@phystech.edu

ABSTRACT

This paper presents a deterministic linear time complexity IDE-SimRank method to approximately compute SimRank with proved error bound. SimRank is a well-known similarity measure between graph vertices which relies on graph topology only and is built on intuition that "two objects are similar if they are related to similar objects". The fixed point equation for direct SimRank computation is the discrete Lyapunov equation with specific diagonal matrix in the right hand side. The proposed method is based on estimation of this diagonal matrix with GMRES and use this estimation to compute single-source and single pairs queries. These computations are executed with the part of series converging to the discrete Lyapunov equation solution.

Keywords

SimRank, graph, similarity measure, Lyapunov equation, inexact GMRES

1. INTRODUCTION

This paper presents a new method to efficiently compute the SimRank [?] which is a topologically induced similarity measure between two given vertices of a graph.

Similarity measures for graphs are useful in many applications such as relation mining [?], document-by-document querying [?, ?] and many others. A major problem in SimRank computation is the high storage and time complexity of the direct iterative process converging to SimRank. Several schemes have been presented for the approximate computation of the SimRank which are based on different concepts [?, ?, ?]. In this paper we propose a two-step method for the approximation of the SimRank. The first and the

most expensive step is the computation where we iteratively estimate the diagonal of the SimRank matrix. After that the SimRank scores can be computed by the approximate solution of the discrete Lyapunov equation. Such approach has been considered in [?]. The difference is that instead of using Gauss-Seidel method combined with Monte-Carlo computations to estimate the diagonal we use numerical linear algebra techniques. We prove that the linear system for the diagonal has bounded condition number, and we have an $\mathcal{O}(n)$ matrix-by-vector product with guaranteed accuracy, thus *inexact GMRES method* is the method of choice. Using the theory of IGMRES we get a provable $\mathcal{O}(n)$ complexity algorithm for the computation of SimRank scores. The final SimRank approximation is a sparse matrix. We also provide an efficient practical algorithm for the computation of SimRank.

2. PROBLEM STATEMENT

Let $G = (V, E)$ be a graph, where V is a set of vertices and E is a set of edges. The order of the graph is the number of vertices $|V| = n$. Similarity measures between graph vertices are very useful in some applications. One of the approach to define such similarity measure is SimRank [?]. In the foundation of SimRank definition lies idea that "two objects are similar if they are referenced by the similar objects". It is proposed that vertex similarity lies between 0 and 1 with vertex being maximally similar to itself with similarity 1. By $s(a, b)$ denote the SimRank between vertices a and b defined as

$$s(a, b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{if } I(a) = \emptyset \text{ or } I(b) = \emptyset \\ \frac{c}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)), & \text{otherwise,} \end{cases} \quad (1)$$

where $I(v)$ is the set of in-neighbours of vertex v , constant $c \in (0, 1)$. Denote by S a SimRank matrix, where (i, j) -th element is SimRank between i -th and j -th vertices. To find the SimRank matrix S , one writes $s(a, b)$ for all pairs (a, b) of vertices and obtains linear system of n^2 equations which has unique solution [?]. Let A be an adjacency matrix of

the graph G normalized by columns:

$$\sum_{i=1}^n A_{ij} = 1, \quad j = 1, \dots, n,$$

and

$$W = \sqrt{c}A.$$

The SimRank matrix S is the solution of the following equation:

$$S = W^\top S W - \text{diag}(W^\top S W) + I = W^\top S W + D, \quad (2)$$

and D is the diagonal matrix $D = -\text{diag}(W^\top S W) + I$. Equation (2) is typically solved by a fixed-point iteration

$$\begin{aligned} S_0 &= I, \\ S_{k+1} &= cA^\top S_k A - c \text{diag}(A^\top S_k A) + I, \end{aligned} \quad (3)$$

where $\text{diag}(P)$ is an operator that maps given matrix P to the diagonal matrix with diagonal entries equal the diagonal entries of the matrix P . The iteration (3) converges if $c < 1$. Direct usage of (3) requires $\mathcal{O}(n^2)$ memory cells and $\mathcal{O}(n^3)$ operations for one iteration, thus is infeasible for real-world graphs. In this paper we propose a new *Iterative Diagonal Estimation method* (IDE-SimRank method) that approximately computes the SimRank in $\mathcal{O}(n)$ time and memory.

3. IDE-SIMRANK METHOD

In this section we describe IDE-SimRank method and give theoretical foundation which proves the reasoning and accuracy of our method.

From (2) it is easy to get a linear system with n unknowns. Since $\text{diag } S = I$ by definition, and

$$S = W^\top S W + D,$$

where D is a diagonal matrix, we get

$$I = \text{diag}(W^\top S W) + D = F(D), \quad (4)$$

where $F(D)$ is a linear operator that maps a diagonal matrix (i.e., a vector of length n) to a diagonal matrix (also a vector of length n), and is defined as

$$F(D) = D + \text{diag}(W^\top S(D)W), \quad (5)$$

and $S(D)$ is the solution of the discrete Lyapunov equation

$$S(D) = W^\top S(D)W + D.$$

So, to evaluate $F(D)$ for a given D we have to solve the discrete Lyapunov equation and take only the diagonal of the solution. This can not be done exactly in $\mathcal{O}(n)$ complexity, but it is possible to do *approximate computation* of $F(D)$ with guaranteed accuracy. Moreover, the operator $F(D)$ is well-conditioned, so it is natural to use *iterative methods with inexact matrix-by-vector products* to solve (4). Inexact GMRES [?] is typically a method of choice. In order to get a working method, we need two components:

1. Estimates for the condition number of the linear operator $F(D)$.
2. Algorithm for the computation of $F(D)$ with a given accuracy ε . (and estimate of its complexity).

Note that the equation (4) was used in the paper [?] under the name *Linearized SimRank*. The difference in our approach is that we use sparse matrix arithmetic and inexact iterative method for its solution (compared to the Gauss-Seidel method combined with Monte-Carlo estimation to compute $S(D)$).

3.1 Estimation of condition number of $F(D)$

Let $\text{vec}(\cdot)$ be an operator that maps an $n \times n$ matrix to a vector of length n^2 taking column-by-column. Denote by $Pv = \text{vec}(D(v))$ an operator that maps a vector v of length n to a vector of length n^2 , where $D(v)$ is a diagonal matrix with v on the diagonal. Now by a slight abuse of notation let F and S act on a vector d of length n . Then, the matrix corresponding to the operator $F(d)$ can be written using Kronecker products as

$$\begin{aligned} F &= I + P^\top (W^\top \otimes W^\top) (I - W^\top \otimes W^\top)^{-1} P \\ &= P^\top (I - W^\top \otimes W^\top)^{-1} P. \end{aligned} \quad (6)$$

The matrix P is the submatrix of the $n^2 \times n^2$ identity matrix, thus F is a submatrix of the matrix $(I - W^\top \otimes W^\top)^{-1}$.

The matrix F is the submatrix of the inverse M -matrix, thus it is also the inverse M -matrix (see [?]), i.e. it is non-singular. Moreover, its condition number can be bounded.

THEOREM 1.

$$\kappa(F)_1 \leq \frac{2(1+c)}{(1-c)^2}.$$

PROOF. For simplicity, introduce the matrix

$$Z = W^\top \otimes W^\top.$$

The matrix Z is nonnegative and $\|Z\|_1 = c$. Since F is a submatrix of the matrix $(I - Z)^{-1}$, there exists an $n^2 \times n^2$ permutation matrix Q such that

$$Q(I - Z)Q^\top = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

and

$$Q(I - Z)^{-1}Q^\top = \begin{bmatrix} * & * \\ * & F \end{bmatrix}.$$

Using well-known formulas for block matrix inversion, the matrix F^{-1} can be written as the Schur complement

$$F^{-1} = D - BA^{-1}C,$$

Consequently, the 1-norm of F^{-1} is bounded in the following way:

$$\|F^{-1}\|_1 \leq \|D\|_1 + \|B\|_1 \|A^{-1}\|_1 \|C\|_1$$

The matrices D , B , C are submatrices of the matrix $I - Z$, therefore their norms are bounded by $\|I - Z\|_1 \leq (1+c)$ (the norms of the submatrices can not exceed the norm of the matrix). To estimate $\|A^{-1}\|_1$ note that A is also a principal submatrix of $(I - Z)$, thus it can be represented as

$$(I - \hat{Z}),$$

where $\|\hat{Z}\|_1 \leq c$, therefore using the standard Neumann series argument

$$\|A^{-1}\|_1 \leq \frac{1}{1-c}.$$

Finally,

$$\|F^{-1}\|_1 \leq (1+c) + \frac{(1+c)^2}{1-c} = \frac{2(1+c)}{1-c}.$$

The matrix F is the submatrix of $(I-Z)^{-1}$, thus

$$\|F\|_1 \leq \|(I-Z)^{-1}\|_1 \leq \frac{1}{1-c},$$

and this completes the proof. \square

3.2 Fast approximate matrix-by-vector product

The key component for the efficient solution of the system (4) is the fast evaluation of $F(D)$ for a given D . The main computational cost comes from the solution of the discrete Lyapunov equation of the form

$$S = W^\top SW + D,$$

where D is a diagonal matrix. The solution can be written as

$$S = \sum_{k=0}^{\infty} (W^\top)^k DW^k = \sum_{k=0}^K (W^\top)^k DW^k + R_K = S_K + R_K, \quad (7)$$

where $\|R_K\|_1 \leq c^K$. The truncated series S_K gives an approximation to $S(D)$ with guaranteed accuracy.

Algorithm 1 presents fast approximate matvec algorithm used further in GMRES. In this algorithm the operator $\text{diagonal}(\cdot)$ maps given $n \times n$ matrix to its $n \times 1$ diagonal. Note, that to get $\mathcal{O}(n)$ complexity we have introduced *thresholding*: the elements smaller than τ are zeroed out, and all computations are implement in sparse matrix arithmetic.

Algorithm 1: Fast approximate matvec algorithm

Data: Scaled adjacency $n \times n$ matrix W , given $n \times 1$ vector x , threshold τ , number of iterations K .

Result: Approximate result of matvec y

```

1  $y_0 = x$ 
2  $X_0 = \text{diag}(x)$ 
3 for  $k = 1 \dots K$  do
4    $X_k = W^\top X_{k-1} W$ 
5    $d = \text{diagonal}(X_k)$ 
6   Threshold to zero all elements of  $d$ , which absolute
   values are less than given threshold  $\tau$ .
7    $y_k = y_{k-1} + d$ 
8 end
9  $y = y_K$ 
```

The error of the matrix-by-vector product can be estimated by the following theorem.

THEOREM 2. *The result of Algorithm 1 satisfies*

$$\|y - y_K\|_\infty \leq \tau \frac{(1+c)^K - 1}{c} + c^K. \quad (8)$$

PROOF. Suppose that \hat{X}_k is the result of Algorithm 1 for $\tau = 0$ after k steps, and

$$X_k = \hat{X}_k + E_k, \quad \|E_k\|_C = \delta_k.$$

Then,

$$\delta_{k+1} \leq \delta_k + c\delta_k + \tau.$$

It is obvious that

$$\delta_k \leq \eta_k,$$

where η_k solves

$$\eta_{k+1} = (1+c)\eta_k + \tau,$$

which can be solved as

$$\eta_k = \tau + \frac{((1+c)^k - 1)}{c}.$$

The final result is obtained by using a well-known estimate on the remainder of the Neumann series. \square

It is easy to get the upper bound on the complexity. The number of terms in the SimRank series to get the accuracy ε can be estimated as $\log_c \varepsilon^{-1}$. At each step, the diagonal of the matrix $(W^\top)^k DW^k$ has to be computed. Let m be an average degree of the vertex. Then the sparsity of W^k is bounded by m^k , and the evaluation reduces to the evaluation of the column norms of the matrix $W^k D^{1/2}$. In practice, however, this bound is a significant overestimation.

3.3 Putting it all together

The GMRES algorithm is summarized in Algorithm 2 [?], and it is assumed that the matrix-by-vector product is exact. All other operations can be easily implemented in $\mathcal{O}(n)$ complexity.

Algorithm 2: GMRES algorithm for the solution of the linear system

Data: Matrix A , right-hand side b , initial guess x_0 , stopping tolerance ε .

Result: Approximate solution x_m : $\|Ax_m - b\| \leq \varepsilon$

```

1 Start: compute  $r_0 = b - Ax_0$ ,  $v_1 = r_0/\|r_0\|$ ,  $V_1 = v_1$ ,
    $\beta = \|b\|$ .
2 Iterations:
3 Orthogonalize:  $\tilde{v}_{k+1} = Av_k - V_k h_k$ , where  $h_k = V_k^T Av$ .
4 Normalize:  $v_{k+1} = \tilde{v}_{k+1}/\|\tilde{v}_{k+1}\|$ .
5 Update:  $V_{k+1} = (V_k \ v_{k+1})$ ,  $H_k = \begin{bmatrix} H_{k-1} & h_k \\ 0 & \|v_{k+1}\| \end{bmatrix}$ ,
   where the first column in  $H_k$  is omitted when  $k = 1$ .
6 Solve the least squares problem
    $y_k = \arg \min_y \|\beta e_1 - H_k y\|$ .
7  $x_m = x_0 + V_m y_m$ .
8 Restart: compute  $\|r_m\| = \|b - Ax_m\|$ . Stop if  $\|r_m\| \leq \varepsilon$ .
   Otherwise:  $x_0 = x_m$ ,  $v_1 = r_m/\|r_m\|$  and start Iterations
   again.
```

If the matrix-by-vector products are inexact, the following Theorem gives the error bound.

THEOREM 3. [?] *After m steps of the inexact GMRES procedure, the following estimation for norm of approximate and real residues holds:*

$$\|r_m - \tilde{r}_m\| \leq \varepsilon$$

if for any $i \leq m$

$$\|\tilde{E}_i\| \leq \frac{\sigma_m(H_m)}{m\|\tilde{r}_m\|} \varepsilon,$$

where $\sigma_m(H_m)$ is a minimal singular value of the Hessenberg matrix corresponding to GMRES process and \tilde{E}_i is an error corresponding to matvec on the i -th iteration.

4. COMPARISON WITH EXISTING METHODS

SimRank algorithm computes similarities between vertices of the input graph $G = (V, E)$. Here we compute a single-source SimRank and a one-pair SimRank. The single-source SimRank is the vector with SimRank scores between given vertex $a \in V$ and all other vertices $b \in V$. The one-pair SimRank is the similarity measure $s(a, b)$ between two given vertices a and b .

The proposed method has memory requirement $\mathcal{O}(n)$ and computational complexity $\mathcal{O}(n)$ of the pre-computation step. Moreover, we compute the sparse approximation to the full SimRank matrix.

For the readers convenience computational and memory complexities of the previously proposed methods are presented Table 1 and 2.

Table 1: Complexities of the single-pair SimRank algorithms

Paper	Time		Memory
	Query	Precomputation	
[?]	$\mathcal{O}(kN)$	$\mathcal{O}((N + d)n)$	$\mathcal{O}(nN)$
[?]	$\mathcal{O}(r^4)$	$\mathcal{O}(r^4 n^2)$	$\mathcal{O}(n^2 r^2 + r^4)$
[?]	$\mathcal{O}(k E ^2)$	Not needed	$\mathcal{O}(n^2)$
This	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table 2: Complexities of the single-source SimRank algorithms

Paper	Time		Memory
	Query	Precomputation	
[?]	$\mathcal{O}(Rk S)$	$\mathcal{O}(nk(R + PQ))$	$\mathcal{O}(m + nP)$
[?]		$\mathcal{O}(kd^k)$	$\mathcal{O}(d^k)$
This	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Some papers like [?], [?], [?], [?] consider a solution of the exact Sylvester equation without any estimation as SimRank approximation. However, this approach has two fundamental problems. The first problem is that the exact solution of discrete Lyapunov equation is only an approximation to the initial SimRank definition, so this solution leads to additional errors. We treat this problem by estimation of the diagonal item in discrete Lyapunov equation and get the correct approximate discrete Lyapunov equation for SimRank. The second problem is to solve Sylvester equation as proposed in [?] one needs invert adjacency matrix of the graph which is unstable and leads to loss of sparsity. Instead of invert the matrix of linear operator we use iterative method GMRES with fast approximate matvec implementation.

5. NUMERICAL EXPERIMENTS

5.1 Synthetic test

To confirm the $\mathcal{O}(n)$ complexity of the proposed method, we generate random adjacency matrices with fixed number of nonzero elements in every column and compute SimRank for corresponding graphs. The dependence of time on n is shown on Figure 1. The other parameters are threshold $\tau = 10^{-4}$, the scale parameter $c = 0.6$ and number of iteration $K = 50$. Here nnz is a number of non-zero elements in

every column. The computational cost increases when the adjacency matrix becomes more dense, which is natural.

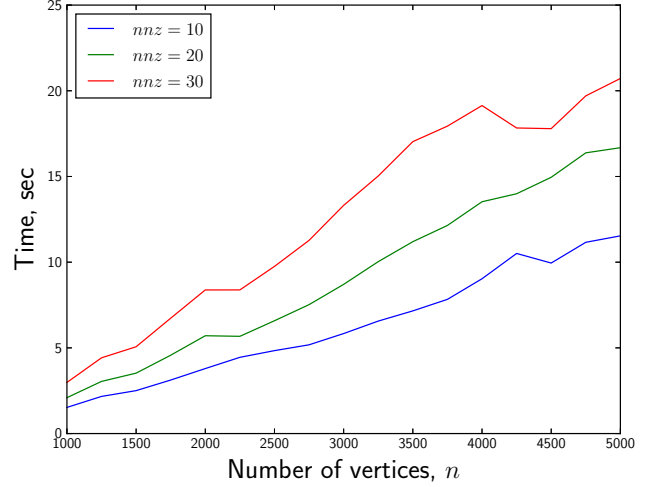


Figure 1: Dependence of time to solve linear system on n for randomly generated graphs with different number of non-zeros

The similar plot for dependence of memory to solve the linear system on the number of vertices n is presented in Figure 2. The other parameters are the same: threshold $\tau = 10^{-4}$, the scale parameter $c = 0.6$ and number of iteration $K = 50$. Here nnz is a number of non-zero elements in every column. The plot shows that the required memory linearly or sub-linearly depends on the number of vertices in the graph. But if the adjacency matrix of the graph is enough sparse, then the required memory is constant and does not depend on the number of vertices.

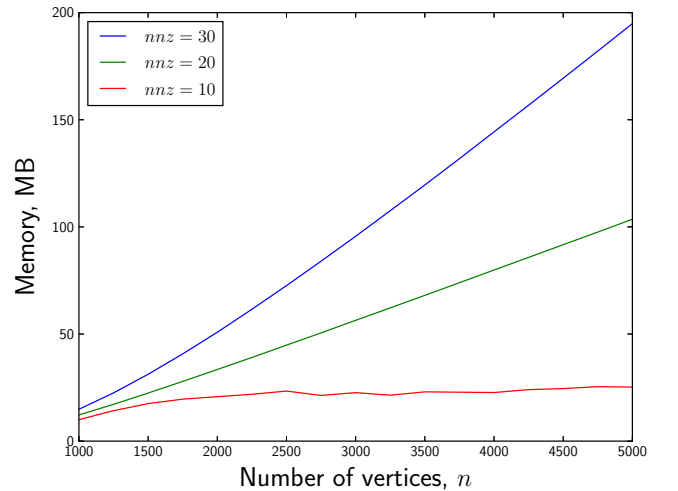


Figure 2: Dependence of memory to solve linear system on n for randomly generated graphs with different number of non-zeros

5.2 DIMACS10 collection

In this section we experimentally study the approximation accuracy of our method. The experiments are carried out on graphs from DIMACS10 Challenge Collection¹. The list of the considered graphs is presented in Table 3.

Table 3: NDCG accuracy for single-source query to the graphs from DIMACS10 Challenge Collection

Name	n	nnz	nnz/n	$1 - \text{NDCG}@n$
chesapeake	39	340	8.72	$1.3 \cdot 10^{-9}$
data	2851	30186	10.59	$3.1 \cdot 10^{-8}$
delaunay n10	1024	6112	5.97	$2 \cdot 10^{-9}$
delaunay n11	2048	12254	5.98	$1.2 \cdot 10^{-8}$
delaunay n12	4096	24528	5.99	$4 \cdot 10^{-8}$
delaunay n13	8192	49094	5.99	$2.5 \cdot 10^{-7}$
uk	4824	13674	2.83	$2.74 \cdot 10^{-7}$
vsp data and seymourl	9167	111732	12.19	10^{-8}

Table 3 presents $1 - \text{NDCG}@n$ measure for convenience. To compute the $\text{NDCG}@n$ measure we make $q = 100$ random queries to SimRank and SimRank approximation for every graph except chesapeake graph ($q = 39$). After that we have two vectors s and \tilde{s} of correct SimRank scores and approximate SimRank scores between query and all other graph vertices. The NDCG measure [?] is defined by the following equation:

$$\text{NDCG} = \frac{1}{Z} \sum_{i=1}^n \frac{2^{\text{rel}_i} - 1}{\log_2(i + 1)},$$

where i is an index of concept, according to the sorted approximate SimRank scores \tilde{s} , $\text{rel}_i = s_i$ is the ground-truth SimRank score between the query and the i -th vertex, and Z is a normalization constant. The other parameters are $c = 0.6$, $K = 50$ and $\tau = 10^{-3}$. Also nnz is the total number of non-zero elements in the adjacency matrix, nnz/n is the average degree of vertex.

5.3 Experiment with Wikipedia

We used Simple English Wikipedia corpus to find semantic relatedness between concepts. The undirected graph corresponding to Simple Wikipedia corpus has 150495 vertices and 4454023 edges. The direct SimRank matrix computation of such large graph is infeasible. Therefore, we assess the quality of our SimRank approximation method not with approximation error but with rationality of the obtained similar concepts. We use the following parameters in the experiment: $c = 0.6$, number of iteration $k = 10$, $\tau = 10^{-4}$.

Table 4 shows some examples of similar concepts extracted from Simple English Wikipedia corpus by the proposed SimRank approximation algorithm. Each column has the queried concept in the top and the most similar concepts to the queried one in the other rows. Every column is sorted according to SimRank scores given by SimRank matrix approximation. We do not display these scores because of the space limitation: the scores differ in 4-th or 5-th significant figures.

6. CONCLUSIONS AND FUTURE WORK

¹<https://www.cise.ufl.edu/research/sparse/matrices/DIMACS10/>

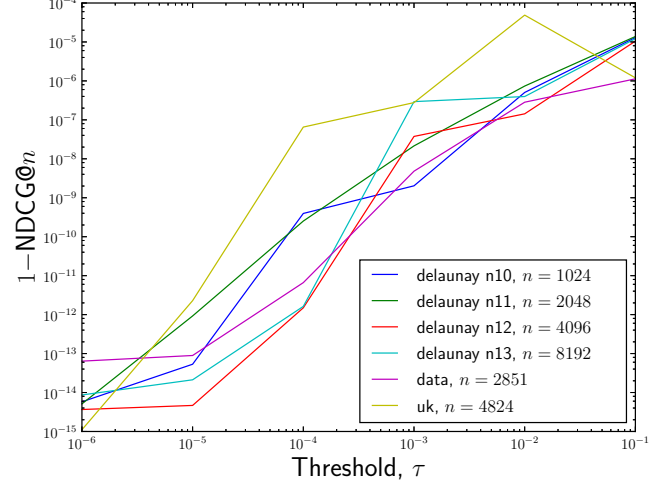


Figure 3: Dependence NDCG on the thresholds for graph from DIMACS10 collection

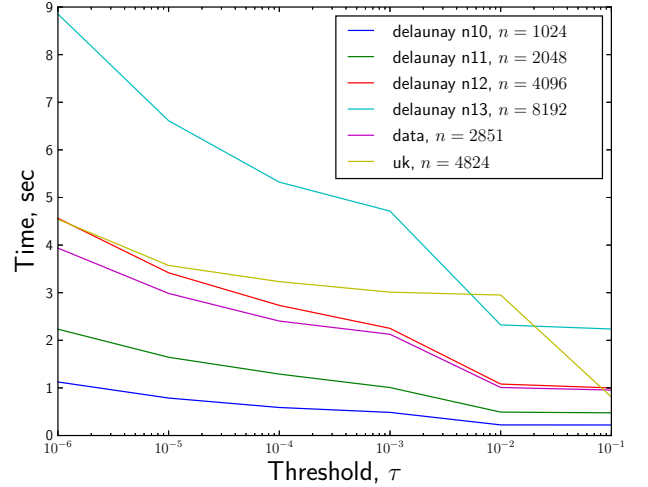


Figure 4: Dependence of time to solve linear system on the thresholds for graph from DIMACS10 collection

An important research direction is the study of hypergraphs, when the adjacency matrix is replaced by the adjacency tensor. We plan to investigate this issue. Also, the computation of the SimRank by summing of the Neumann series can be improved by using more advanced iterative method, like the IGMRES approach considered here, but it requires a lot of technical work.

7. ACKNOWLEDGEMENTS

The authors thank D. Kolesnikov for fruitful discussions.

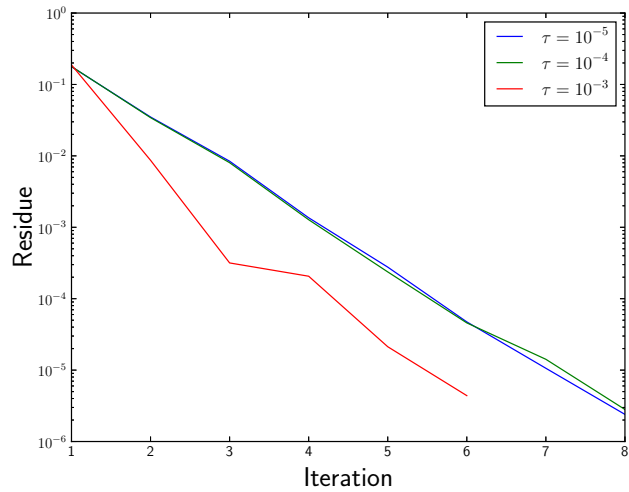


Figure 5: Convergence of the GMRES with different thresholds τ

Table 4: Similar concepts according to proposed SimRank approximation algorithm

<i>GNU</i>	<i>Earth</i>	<i>Liquid</i>
Richard Matthew Stallman	South Pole-Aitken basin	Plasma (matter)
Linux operating system	Frame of reference	Matters
Hurd	Interplanetary internet	Particle theory of matter
Debian linux	Supernova 1987A	Hematological
Linux (kernel)	Probotector	Blude
*nix	Near Earth Object	Human blood